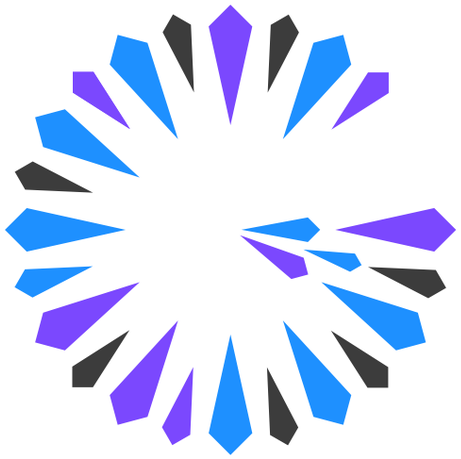


APACHE
GEODE

ActiveMQ Artemis

Scalable IoT Platform

Swapnil Bawaskar

@sbawaskar

sbawaskar@apache.org

Agenda

- Introduction
  - IoT
  - MQTT
  - Apache ActiveMQ Artemis
  - Apache Geode
- Real world use case
- Q&A

IoT

- Devices collect and send data to brokers
- Clients process data to deliver business value
- IoT data platform considerations
  - Protocol
  - How to read
  - How to Analyze
  - How to scale

Protocol

- MQTT
  - Message Queuing Telemetry Transport
  - Based on TCP/IP
  - Optimized binary protocol
  - No type system
  - Provides different QoS levels
  - Low energy consumption

ActiveMQ Artemis

- Subproject of ActiveMQ
- Non blocking architecture
  - High Performance
  - Multi Protocol
  - Embeddable
  - Clustered
- Persistence
  - Journaled
  - Relational database

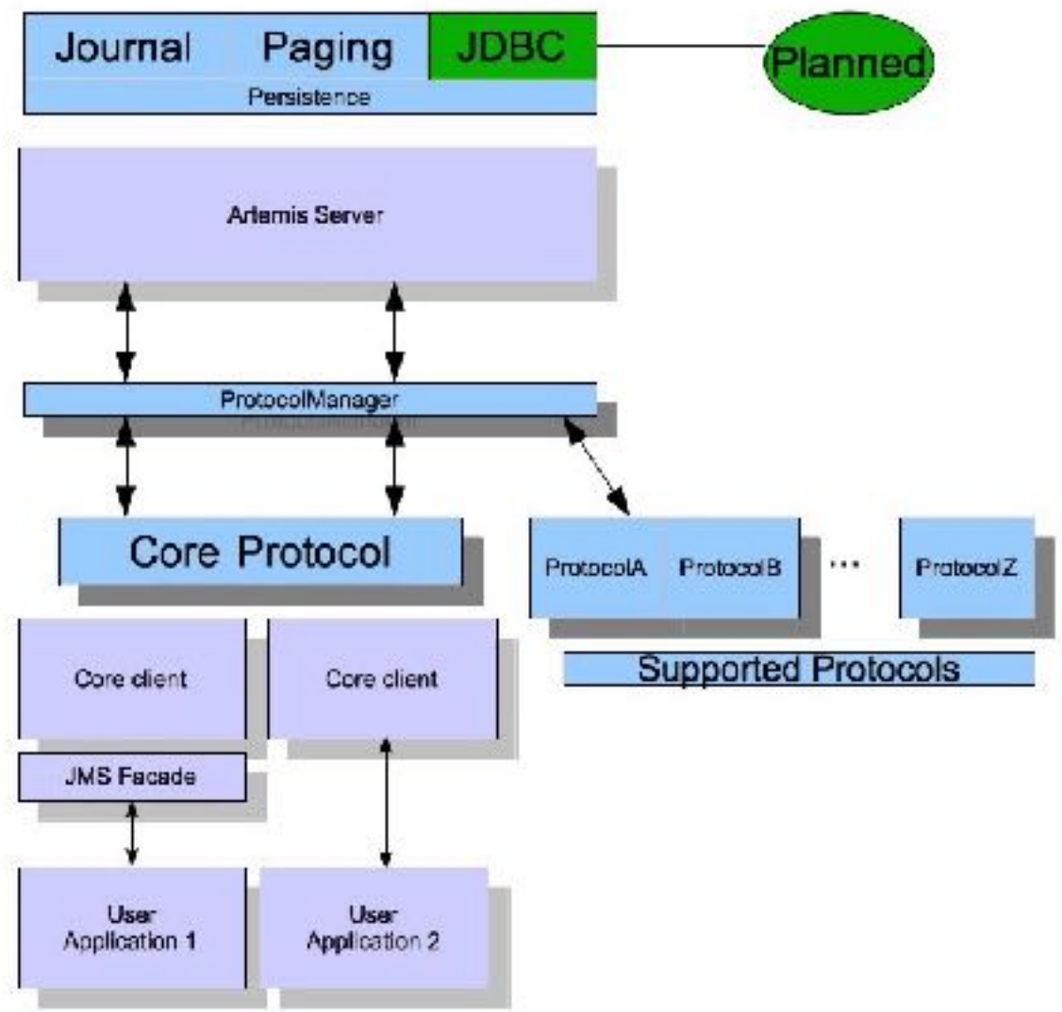
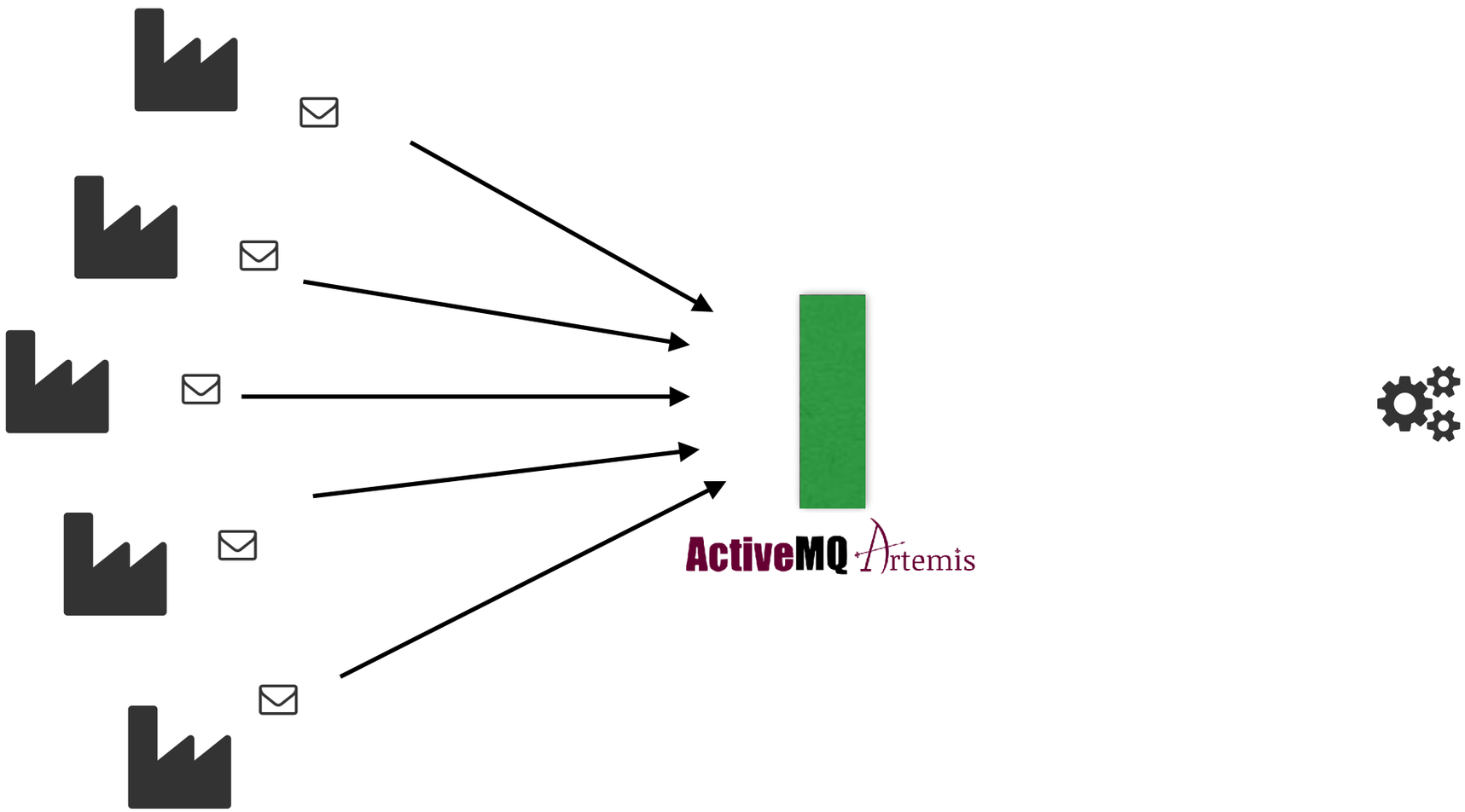


Figure 3.1 Artemis High Level Architecture

ActiveMQ Artemis

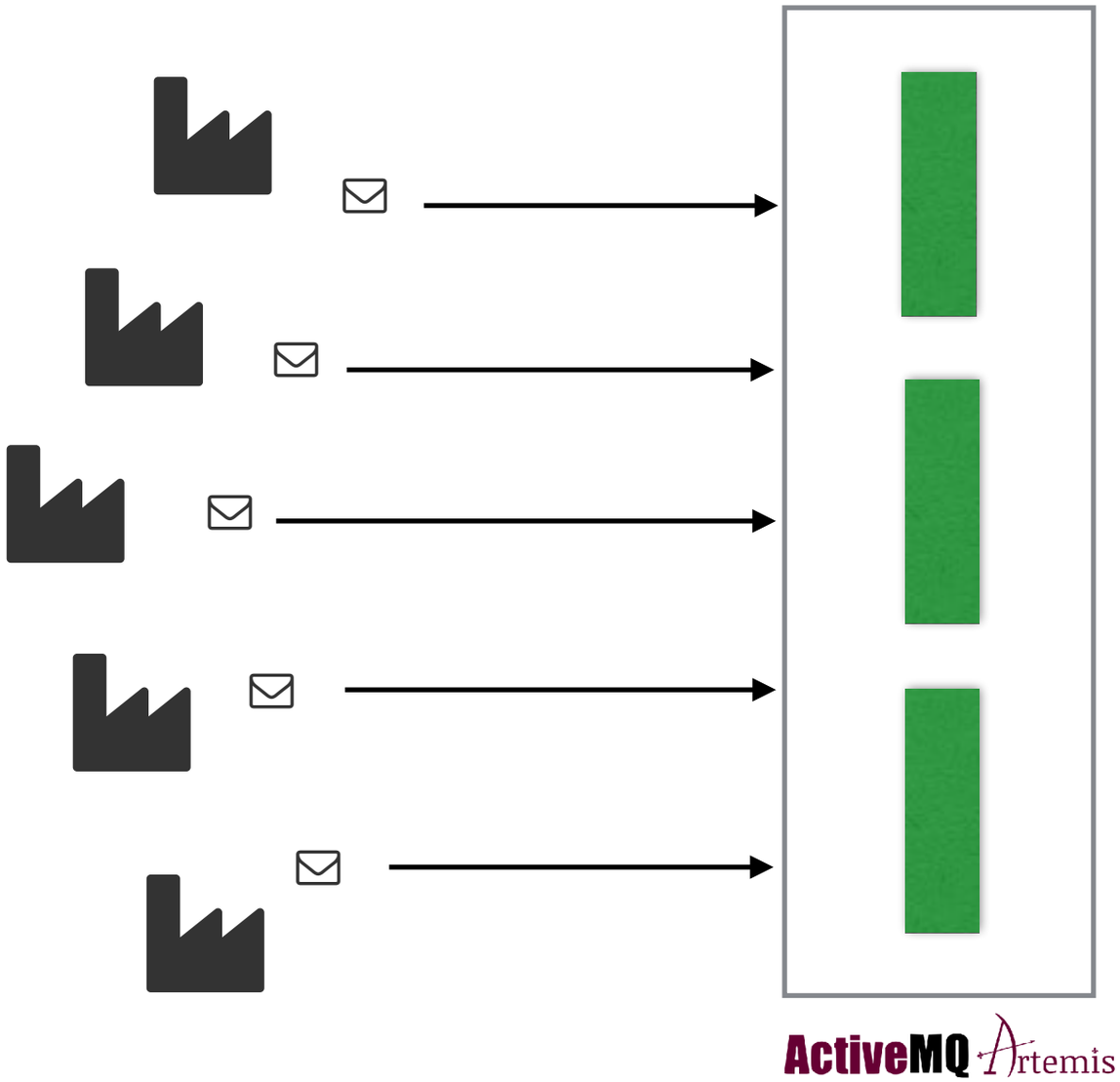
APACHE GEODE

Scaling



- When dealing with large number of devices

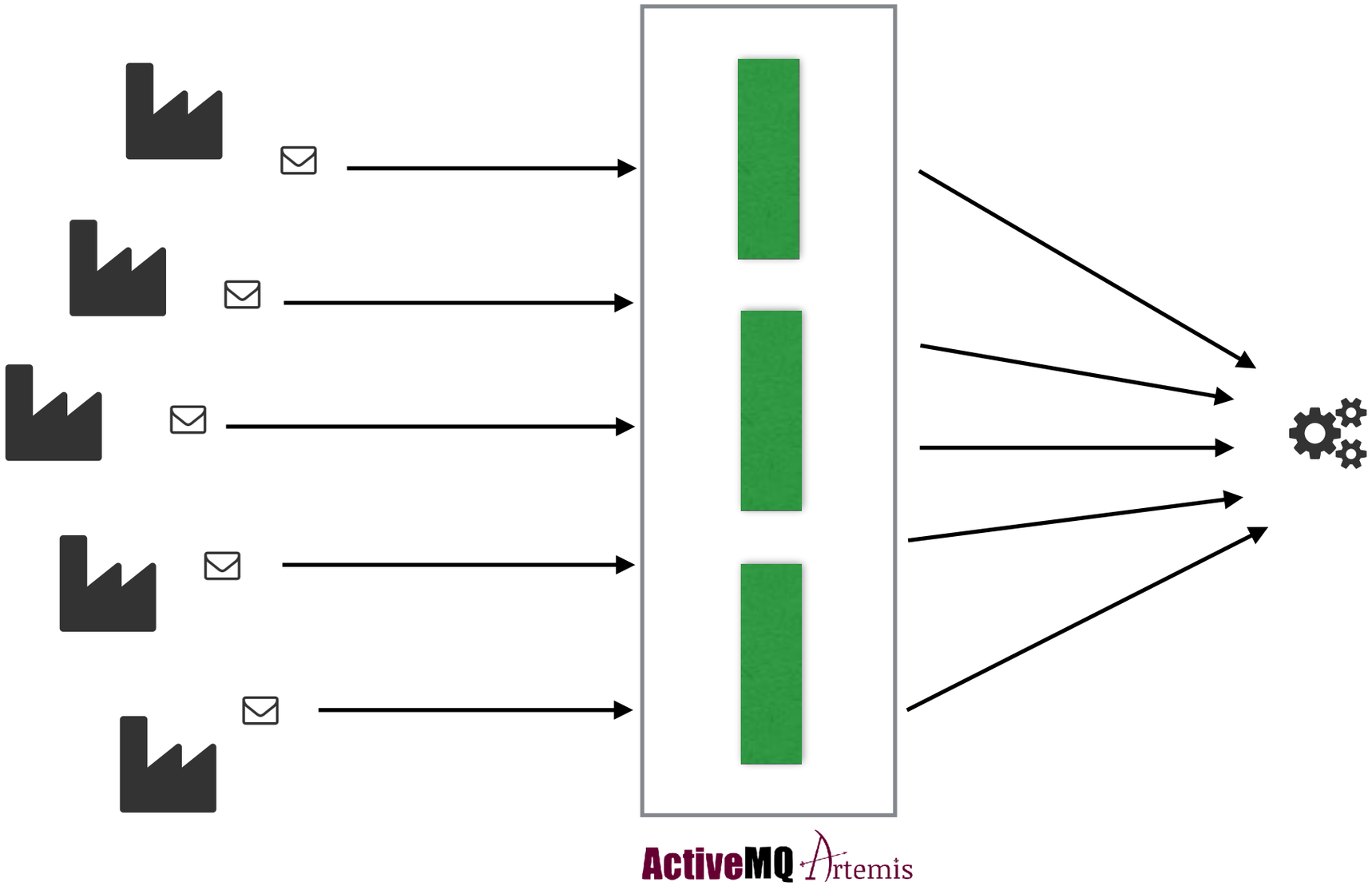
Scaling



Cluster

- Brokers form cluster
- Clients are load balanced

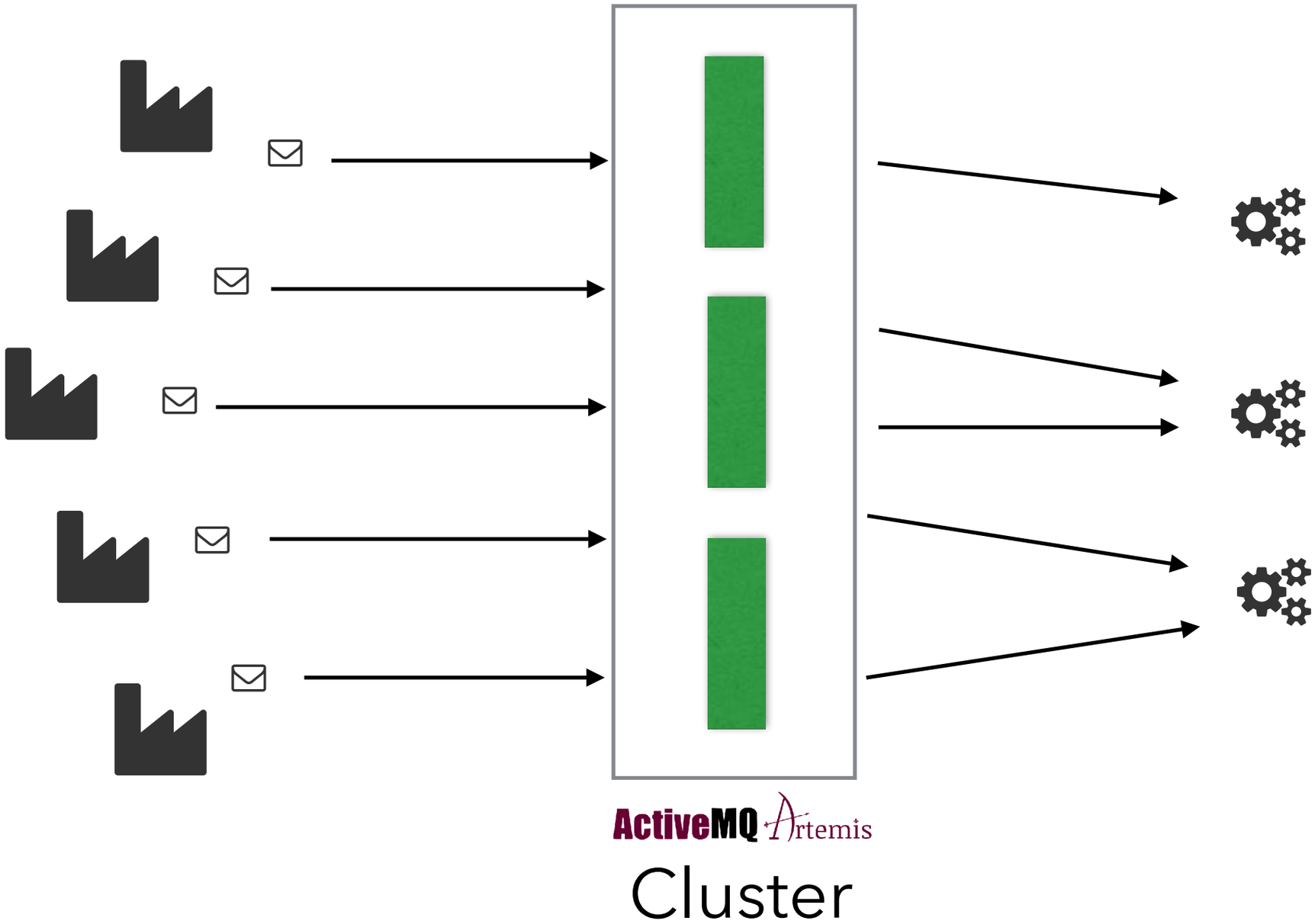
Scaling



Cluster

- Need to scale the processors

Scaling



- Processors do not see all data

Scaling

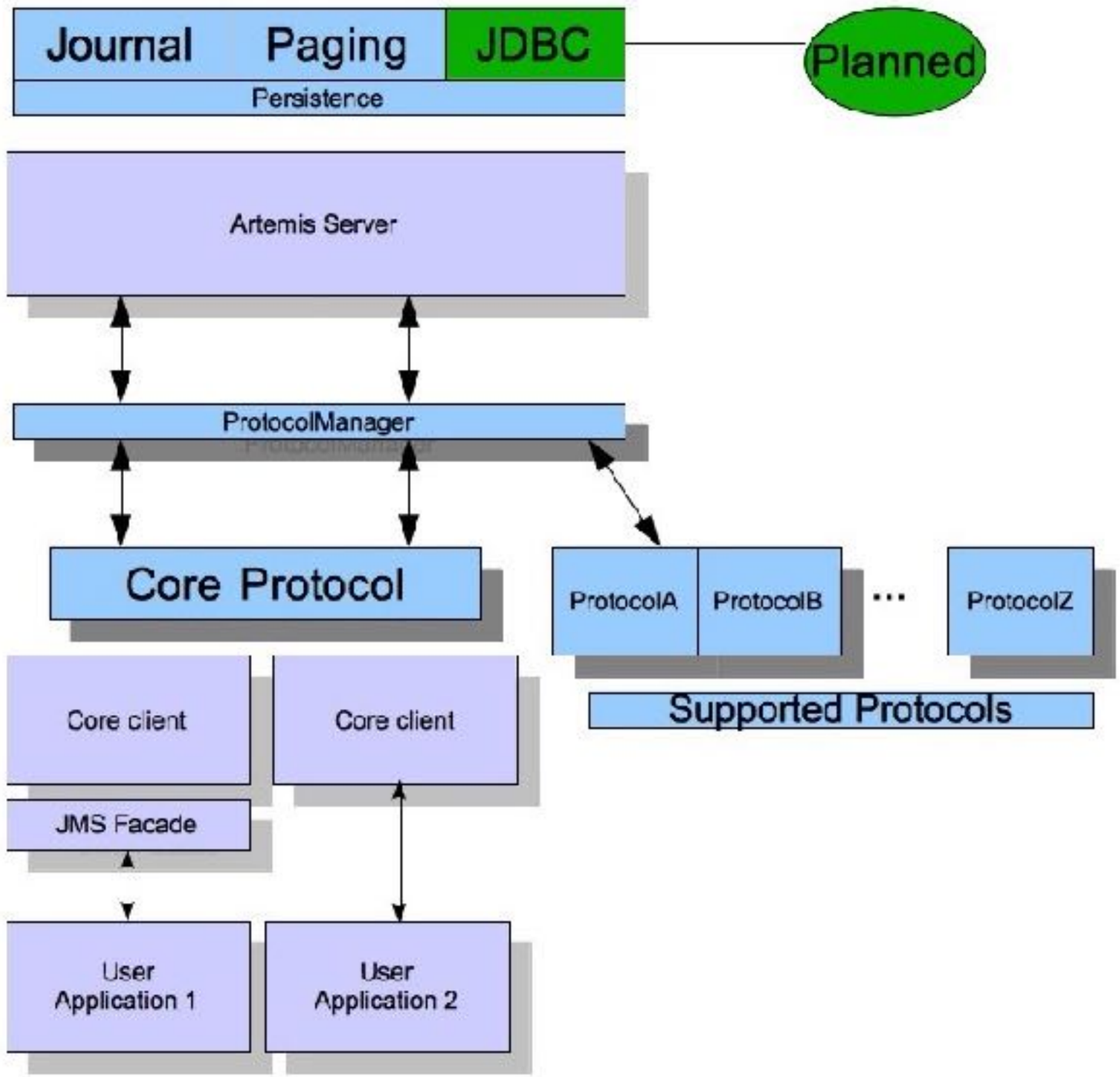


Figure 3.1 Artemis High Level Architecture

Scaling

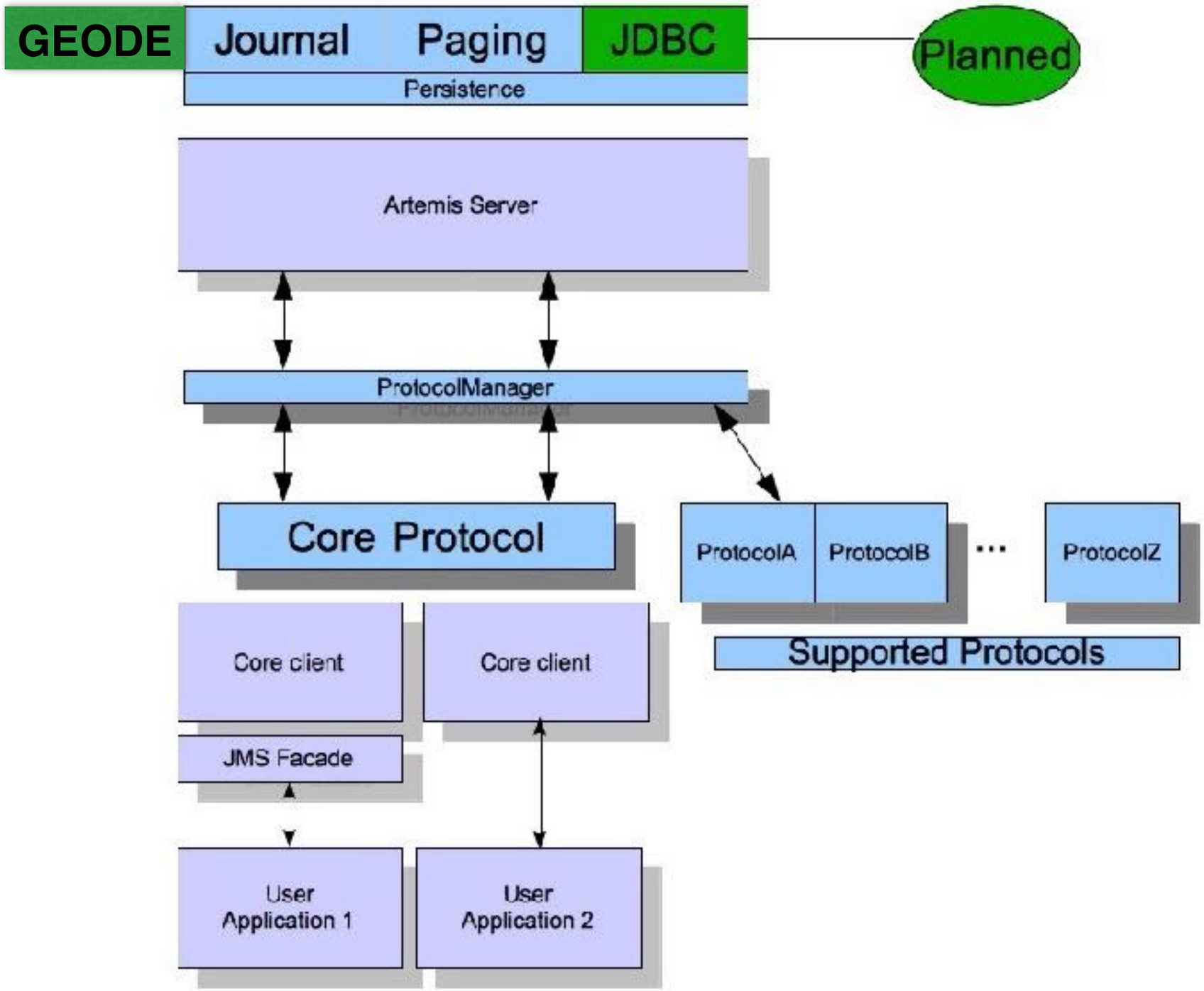
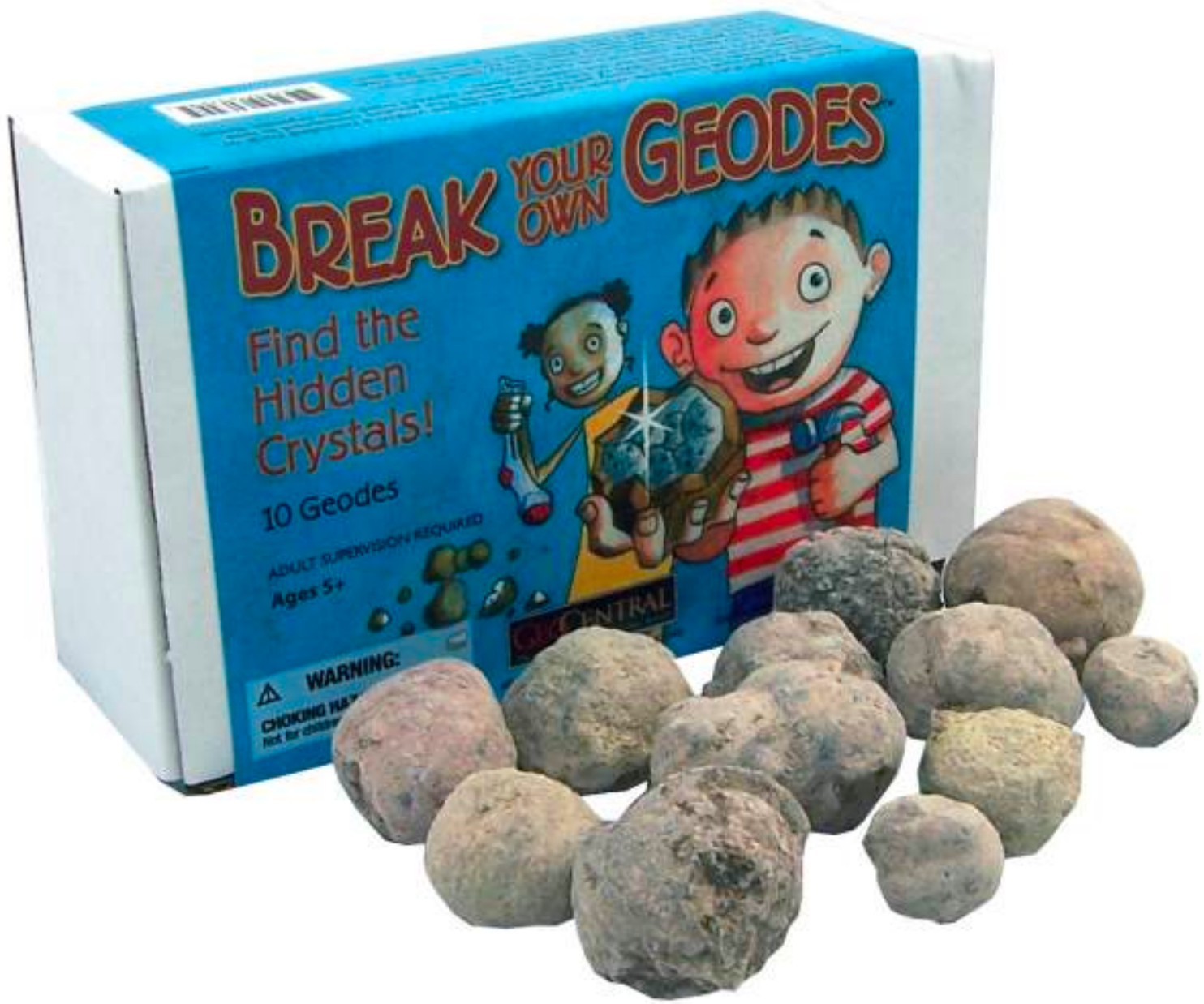


Figure 3.1 Artemis High Level Architecture

What is it?


BREAK YOUR OWN GEODES
Find the Hidden Crystals!
10 Geodes
ADULT SUPERVISION REQUIRED
Ages 5+
WARNING:

What is it?

A distributed, memory-based data management platform for data oriented apps that need:

- high performance, scalability, resiliency and continuous availability
- fast access to critical data set
- location aware distributed data processing
- event driven data architecture

Numbers Everyone Should Know

L1 cache reference	0.5 ns	
Branch mispredict	5 ns	
L2 cache reference	7 ns	
Mutex lock/unlock	100 ns	
Main memory reference	100 ns	
Compress 1K bytes with Zippy	10,000 ns	0.01 ms
Send 1K bytes over 1 Gbps network	10,000 ns	0.01 ms
Read 1 MB sequentially from memory	250,000 ns	0.25 ms
Round trip within same datacenter	500,000 ns	0.5 ms
Disk seek	10,000,000 ns	10 ms
Read 1 MB sequentially from network	10,000,000 ns	10 ms
Read 1 MB sequentially from disk	30,000,000 ns	30 ms
Send packet CA->Netherlands->CA	150,000,000 ns	150 ms

http://static.googleusercontent.com/media/research.google.com/en/us/people/jeff/stanford-295-talk.pdf

Who are the users?

- 17 billion records in memory
  - GE Power & Water's Remote Monitoring & Diagnostics Center
- 3 TB operational data in-memory, 400 TB archived
  - China Railways
- 4.6 Million transactions a day / 40K transactions a second
  - China Railways
- 120,000 Concurrent Users
  - Indian Railways

China Railway Corporation

Indian Railways

Population: 1,401,586,609 1,251,695,616

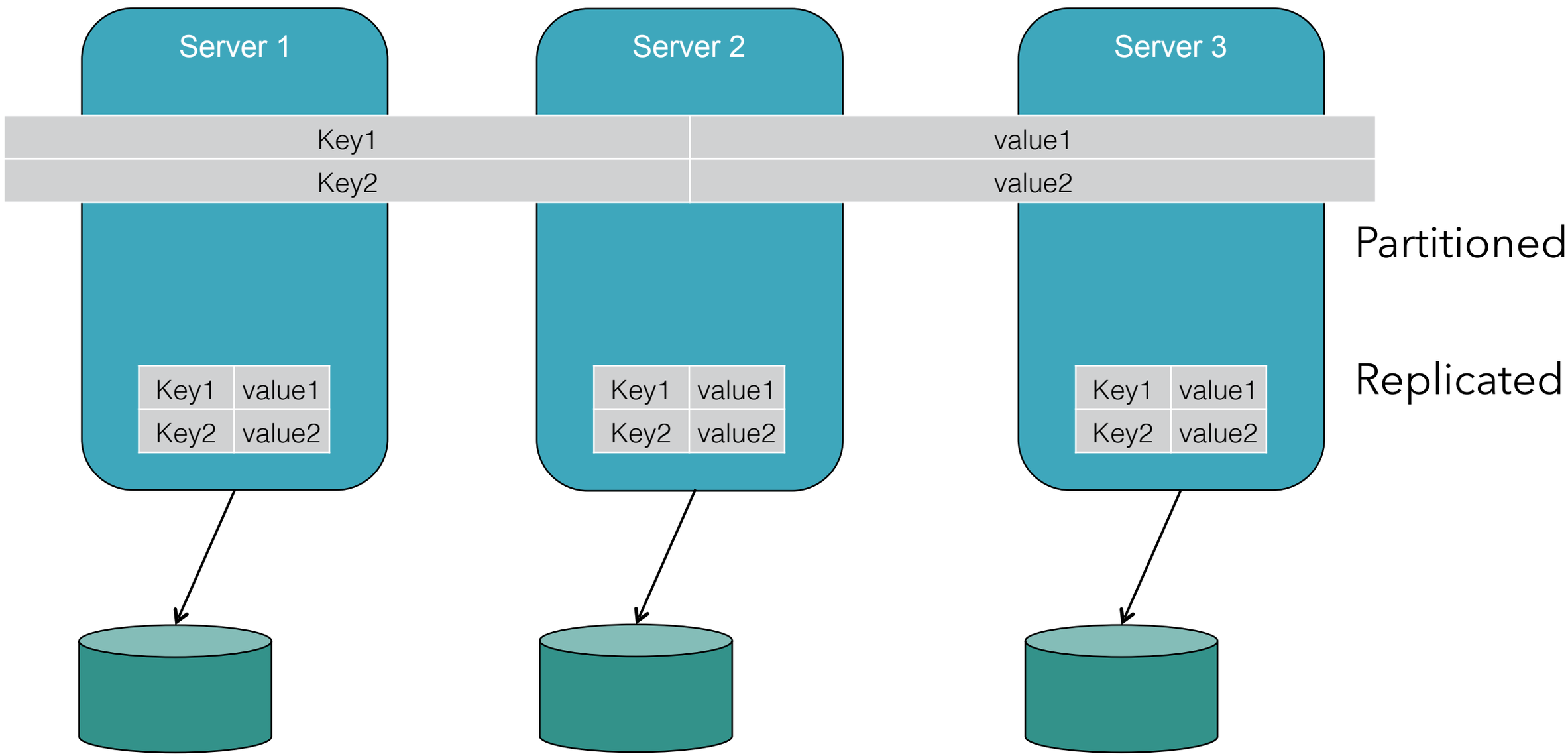
World: ~7,349,000,000

~36% of the world population

APACHE GEODE

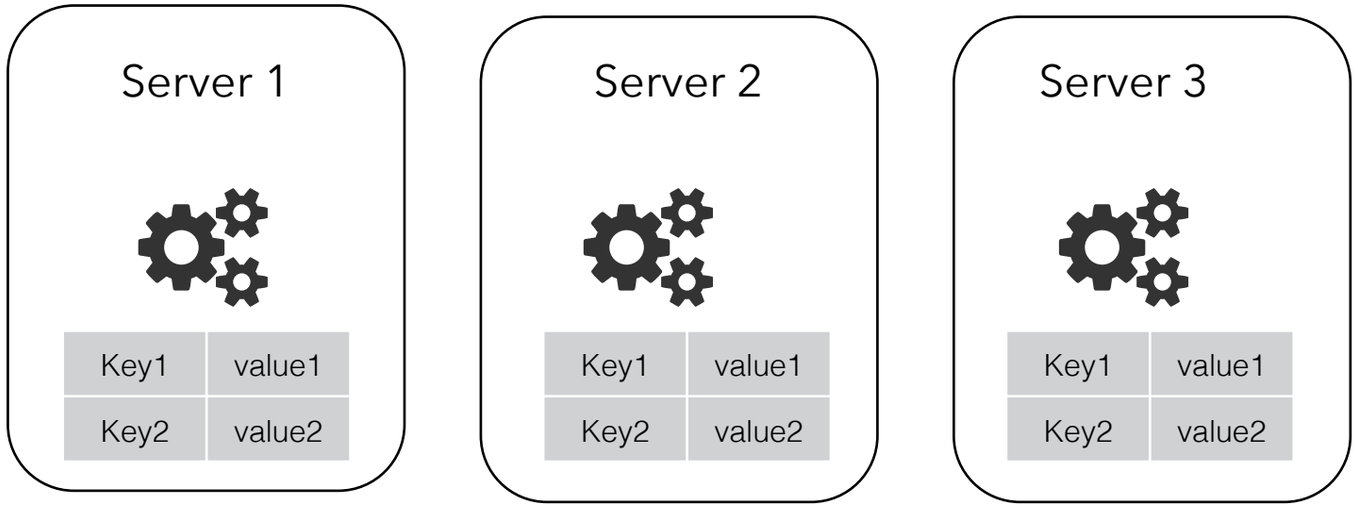
Regions

- Distributed key-value store (`java.util.concurrent.ConcurrentMap`)
- Region due to old JSR-107 spec
- Both Keys as well as Values can be domain objects



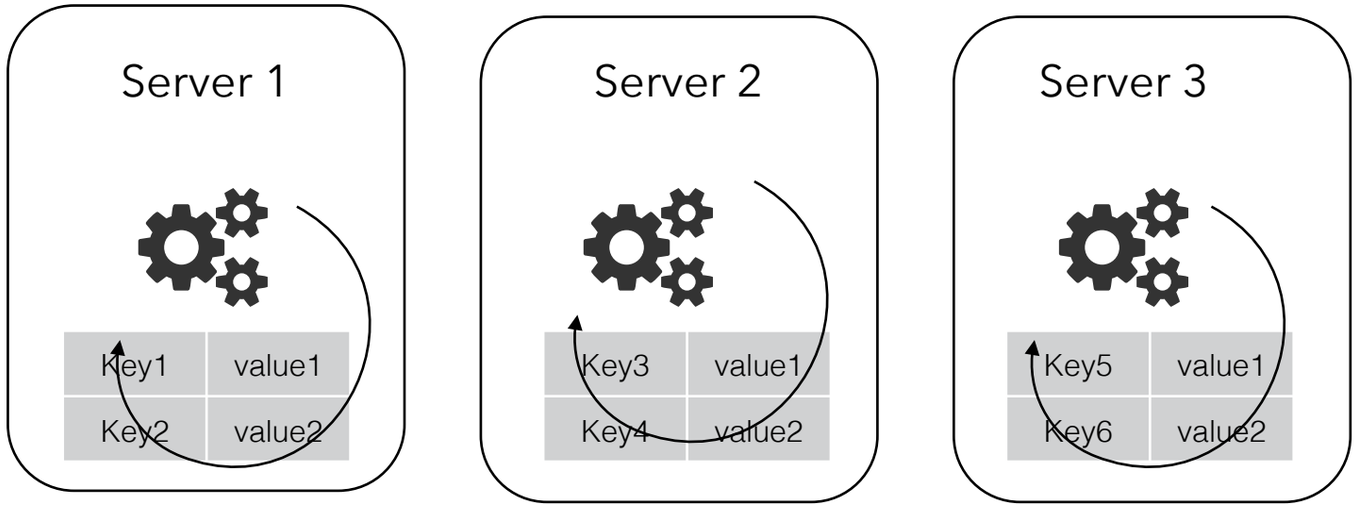
Functions

- Deploy Function on all servers
- Runs in-process with the servers



Functions

- Deploy Function on all servers
- Runs in-process with the servers



Query

- Object Query Language (OQL)
- Similar to SQL
  - SELECT DISTINCT * FROM /exampleRegion WHERE status = 'active'
- You can drill down into domain objects
  - SELECT p.name FROM /person p WHERE p.pet.type='dino'
- You can also invoke methods on your domain objects
  - SELECT DISTINCT * FROM /person p WHERE p.children.size >= 2
- Joins Possible
  - Between Replicate regions
  - Between one Partitioned and Replicate regions
    - SELECT portfolio1.ID, portfolio2.status FROM /exampleRegion portfolio1, /exampleRegion2 portfolio2 WHERE portfolio1.status = portfolio2.status

Continuous Query

- Enables event-driven apps
- Register a Query with the server
  - SELECT * FROM /tradeOrder t WHERE t.symbol='VMW' AND t.price > 100.00
- The server then notifies when the query condition is met
  - Client implements the `CqListener` callback
- HA support
- Domain objects **not** required on the server's class-path

Fixed or flexible schema?

id	name	age	pet_id

or

```
{
  id   : 1,
  name : “Fred”,
  age  : 42,
  pet  : {
    name : “Barney”,
    type : “dino”
  }
}
```

C#, C++, Java, JSON

Portable Data eXchange

```
|              header              |        data        |
| pdx | length | dsid | typeid | fields | offsets |
```

No IDL, no schemas, no hand-coding
Schema evolution (Forward and Backward Compatible)

* domain object classes not required
No need to bring down cluster when domain objects change

APACHE GEODE

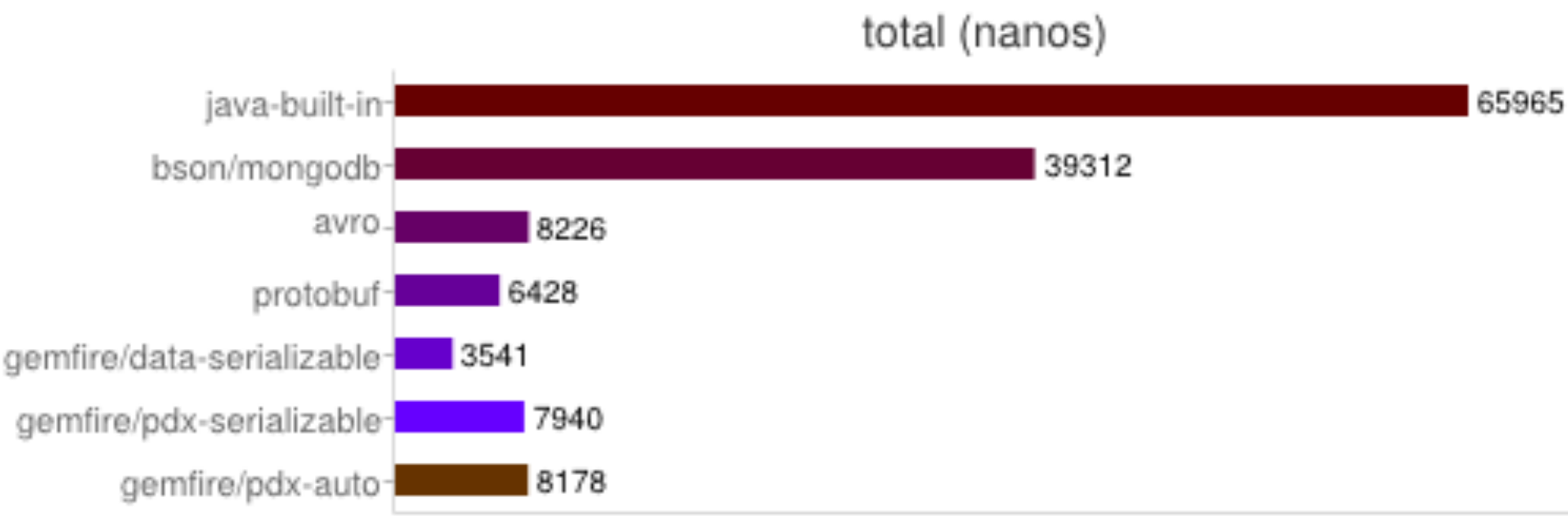
Efficient for queries

```
SELECT p.name FROM /Person p WHERE p.pet.type = “dino”
```

```
{
  id   : 1,
  name : “Fred”,
  age  : 42,
  pet  : {
    name : “Barney”,
    type : “dino”
  }
}
```

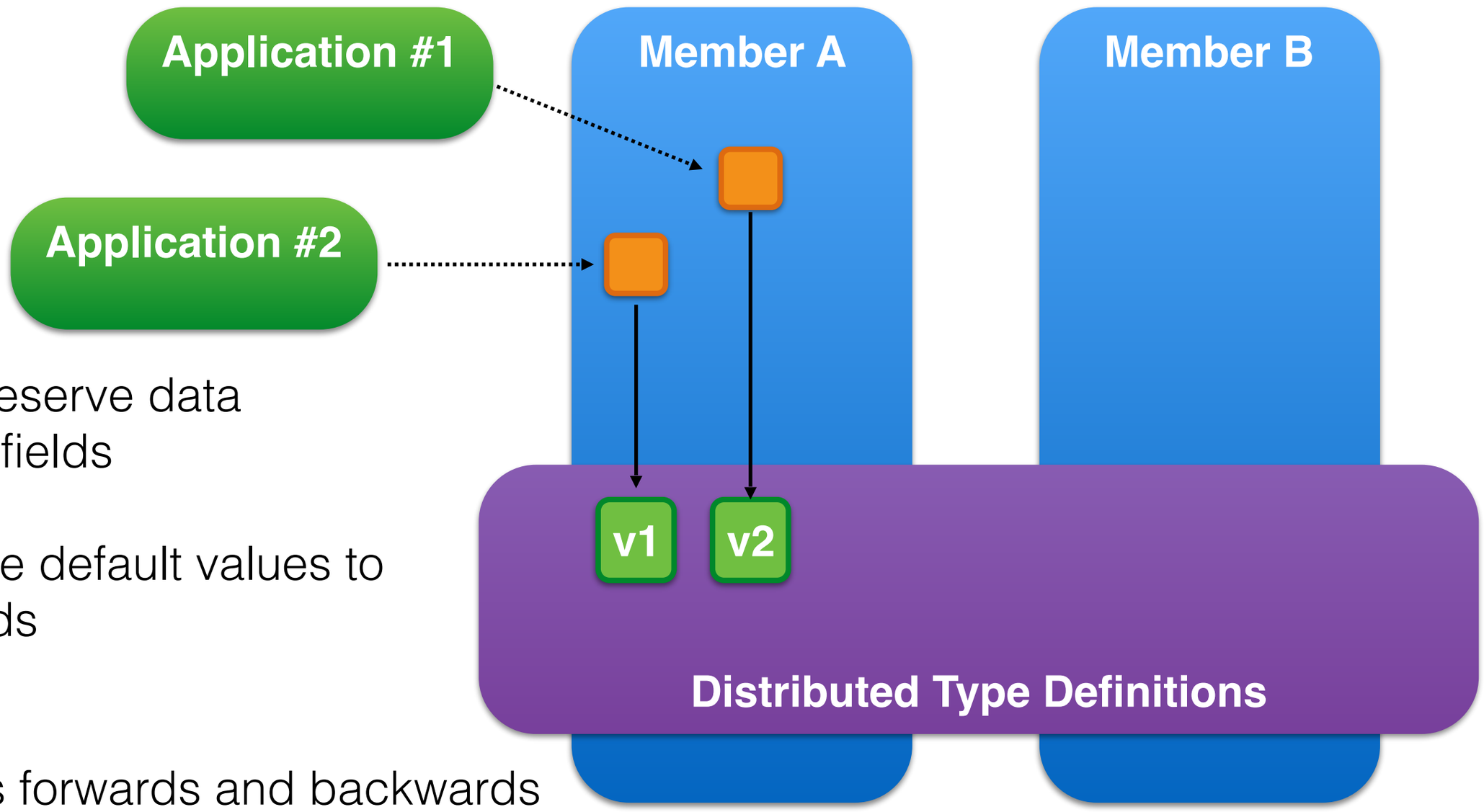
single field
deserialization

But how fast is it?



Benchmark: https://github.com/eishay/jvm-serializers

Schema evolution



v2 objects preserve data from missing fields

v1 objects use default values to fill in new fields

PDX provides forwards and backwards compatibility, no code required

IoT Use Case

- Telemetry data from machines
- Predicting failure
  - Outside one standard deviation
  - Evaluating markov model
- Use functions to iterate over data
- Use CQs to notify
- Update CQs based on function results

Questions?